Session Code: WSV 404

web/services

“Indigo”: The Web Services Protocols And Architecture

Omri Gazitt
Product Unit Manager
Advanced Web Services
omrig@microsoft.com
www.gazitt.com/OhmBlog

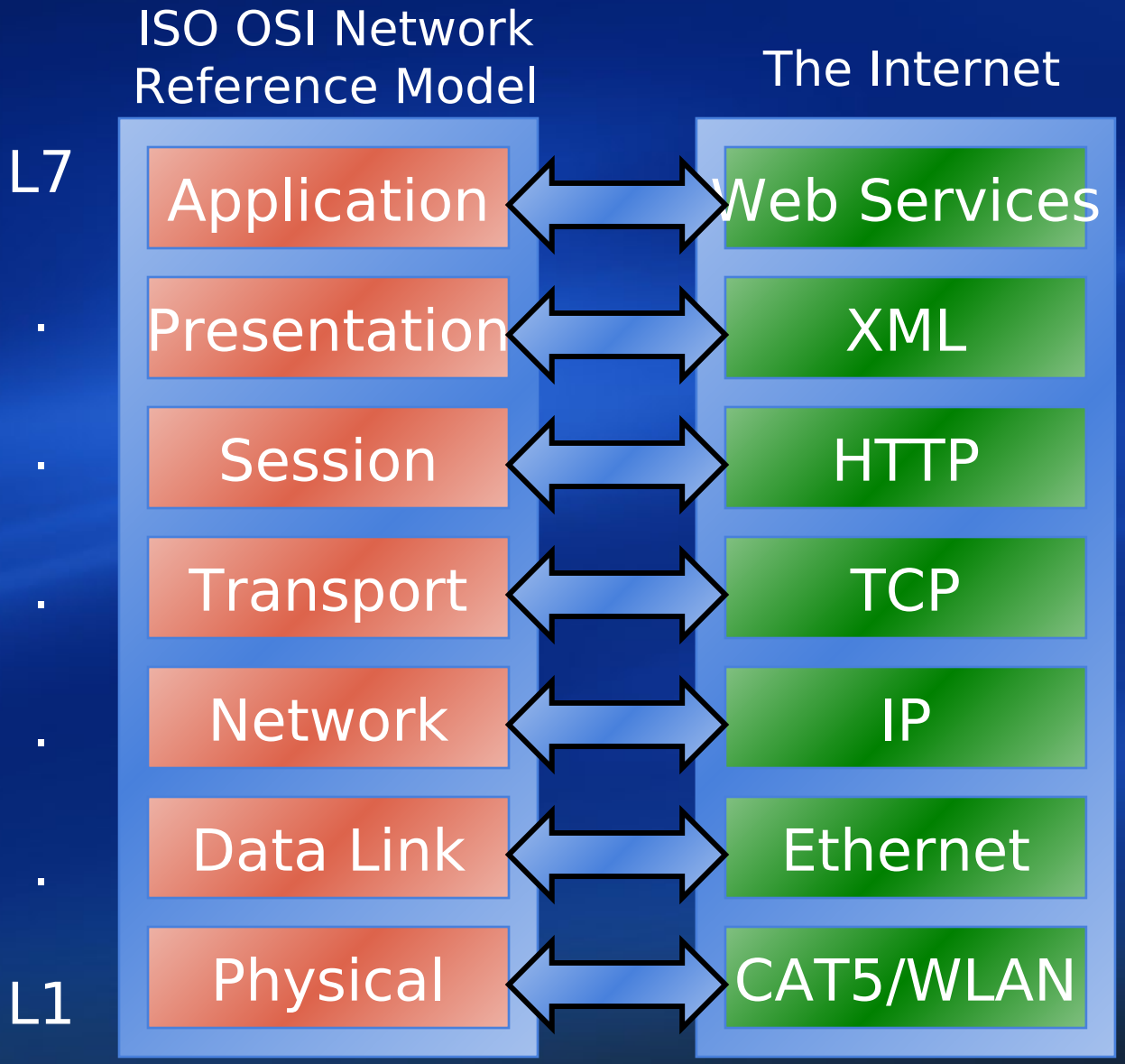
PDC03
Make the connection

Microsoft

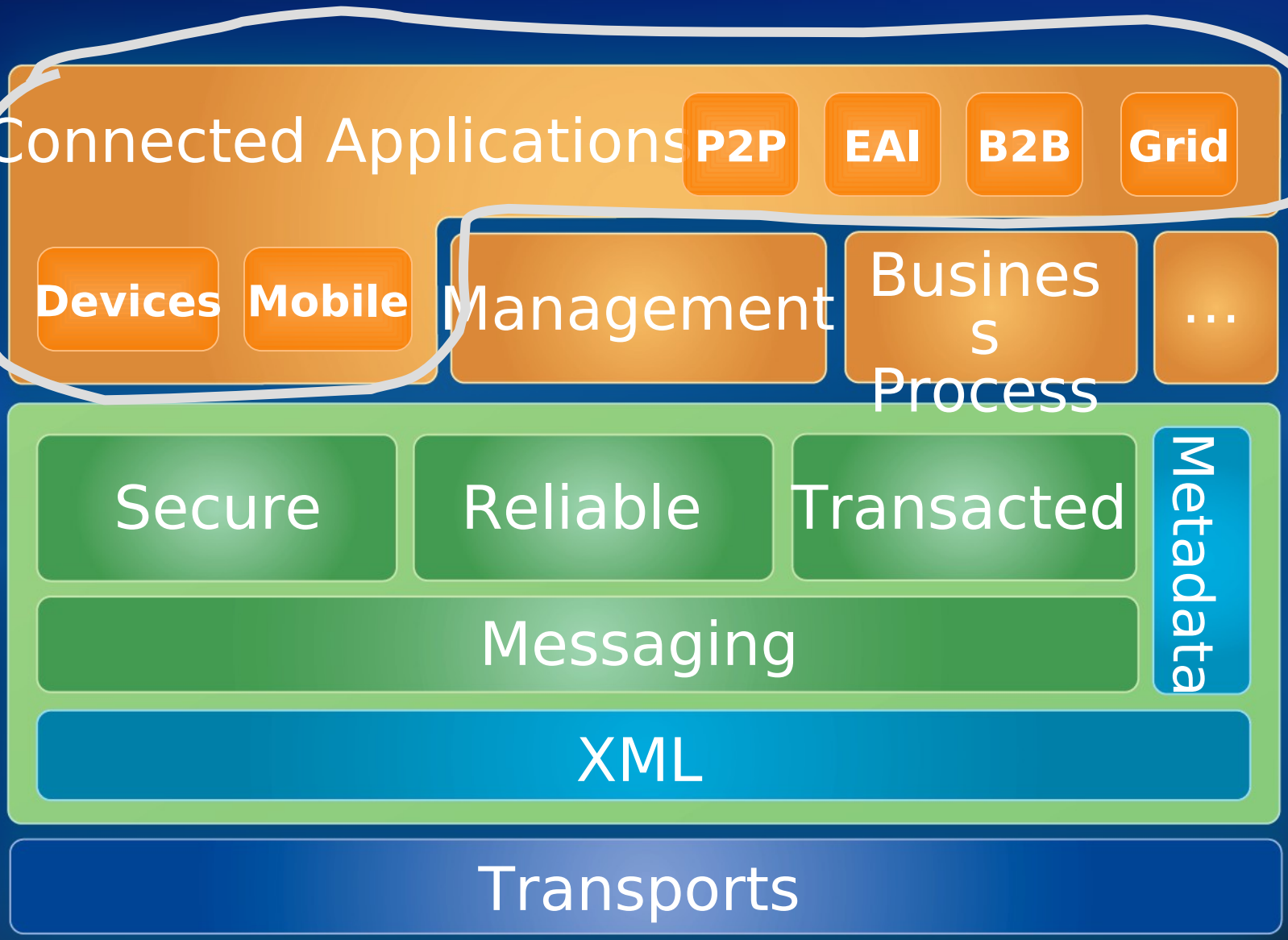
Services And Protocols

- Service-orientation: Crossing boundaries
- Web Services are the Internet’s L7
- IP is on a tear in the device space
  - Web Services will soon follow
- Mobile operators trying to make their networks programmable
- "Indigo" sits at the intersection of these industry trends

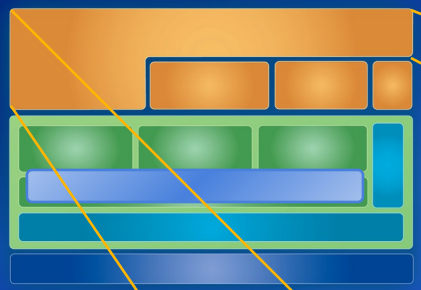
Web Services – Internet's L7



Web Services Architecture



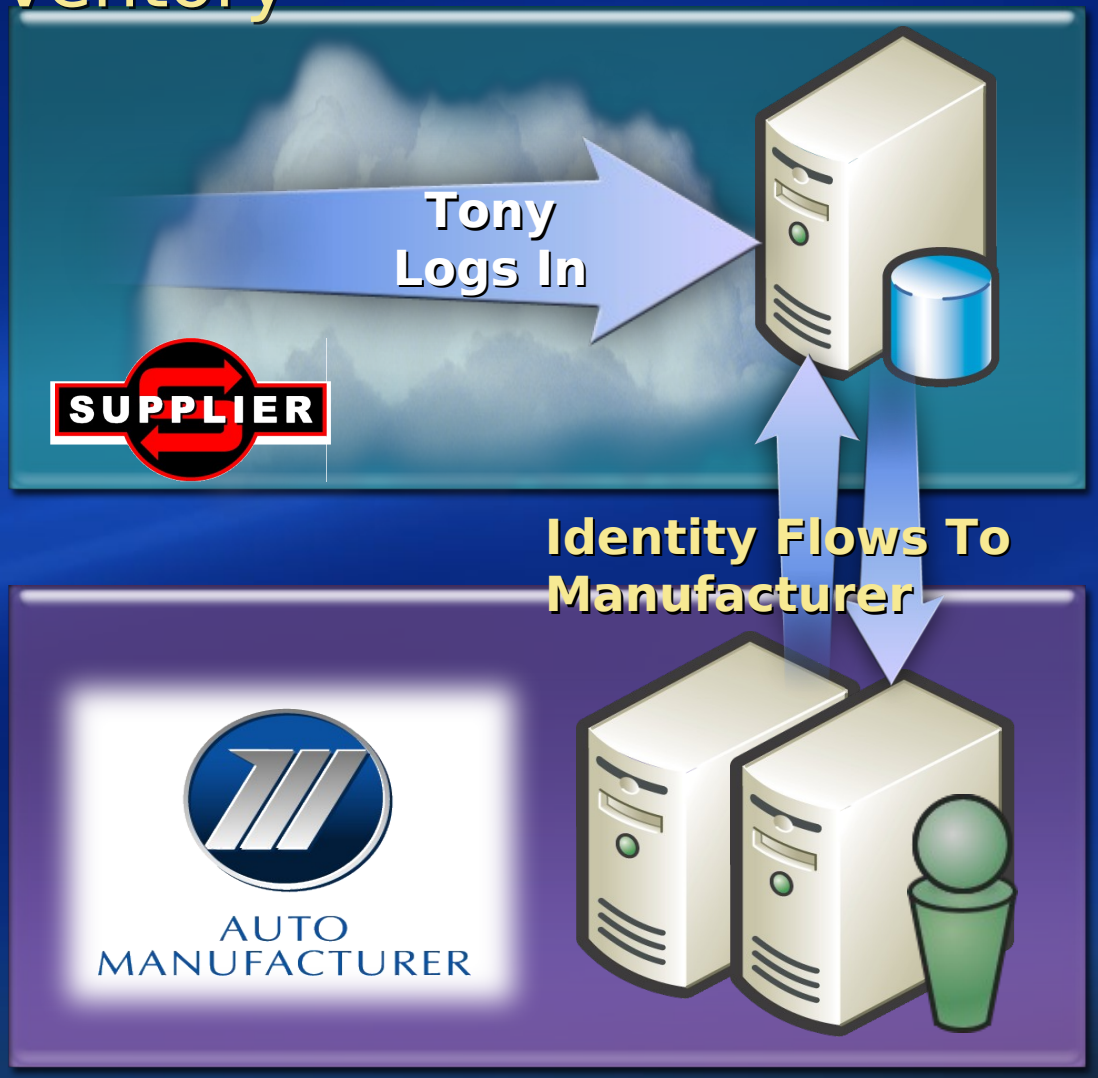
SRT Demo (Gates/Mills, 9/17)



Secure, Reliable, Transacted Supply Chain Management Application

Example Scenario

Roaming Smart Client – Vendor Managed Inventory



Technologies

- Security
- Federation

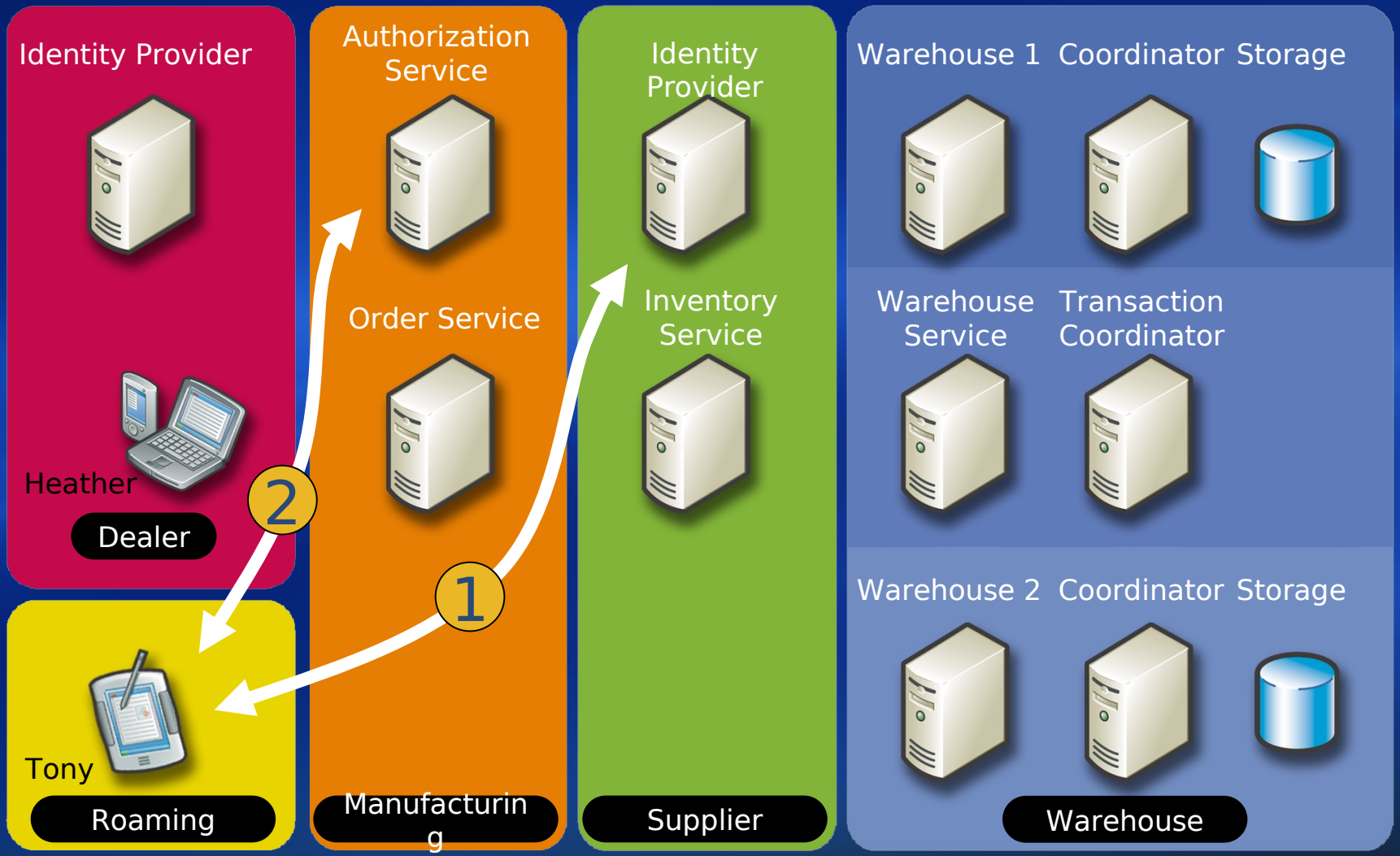
Example Scenario

Roaming Smart Client – Vendor Managed Inventory

Technologies

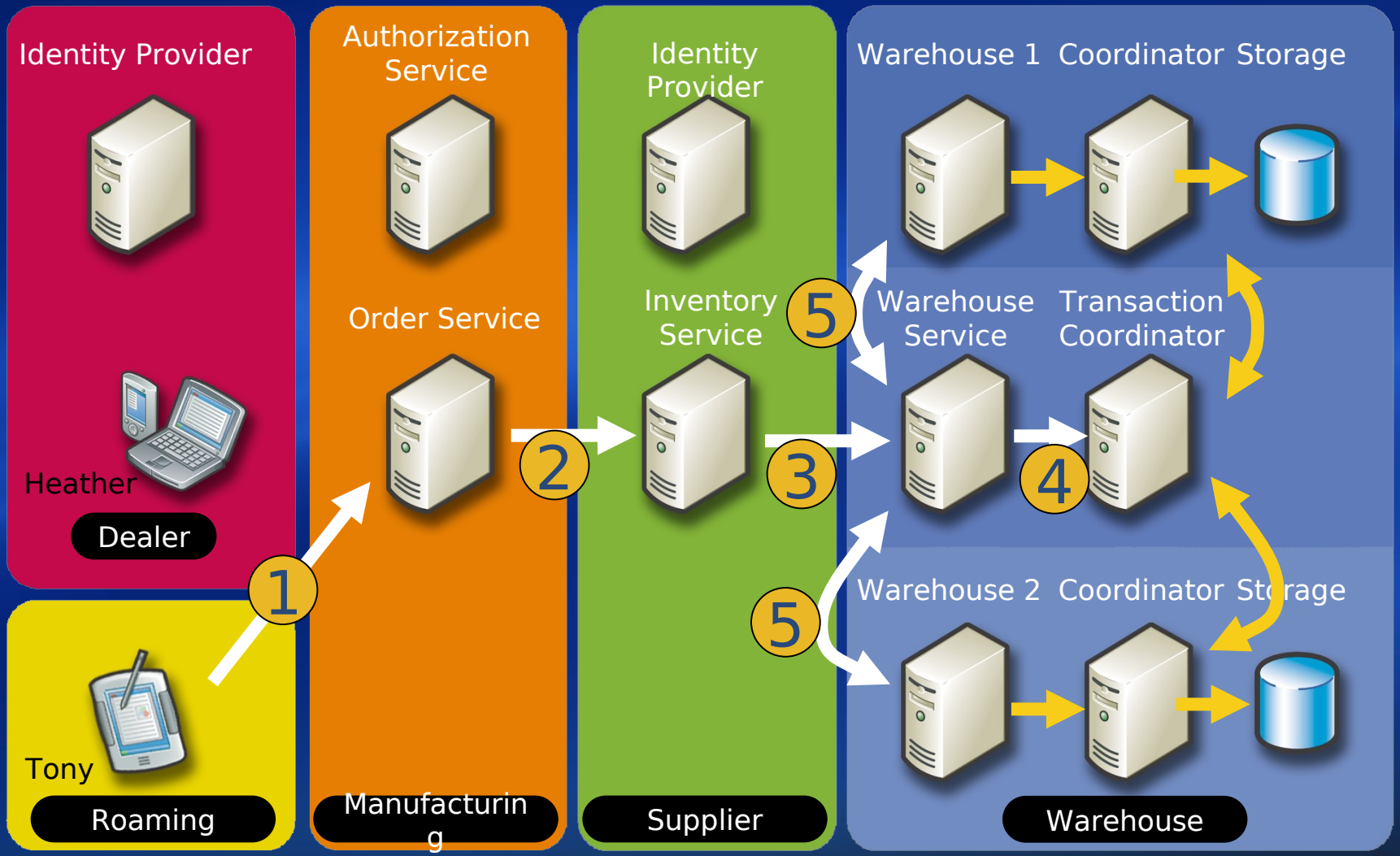
- Security
- Reliability
- Transactions

Tony – Example Login

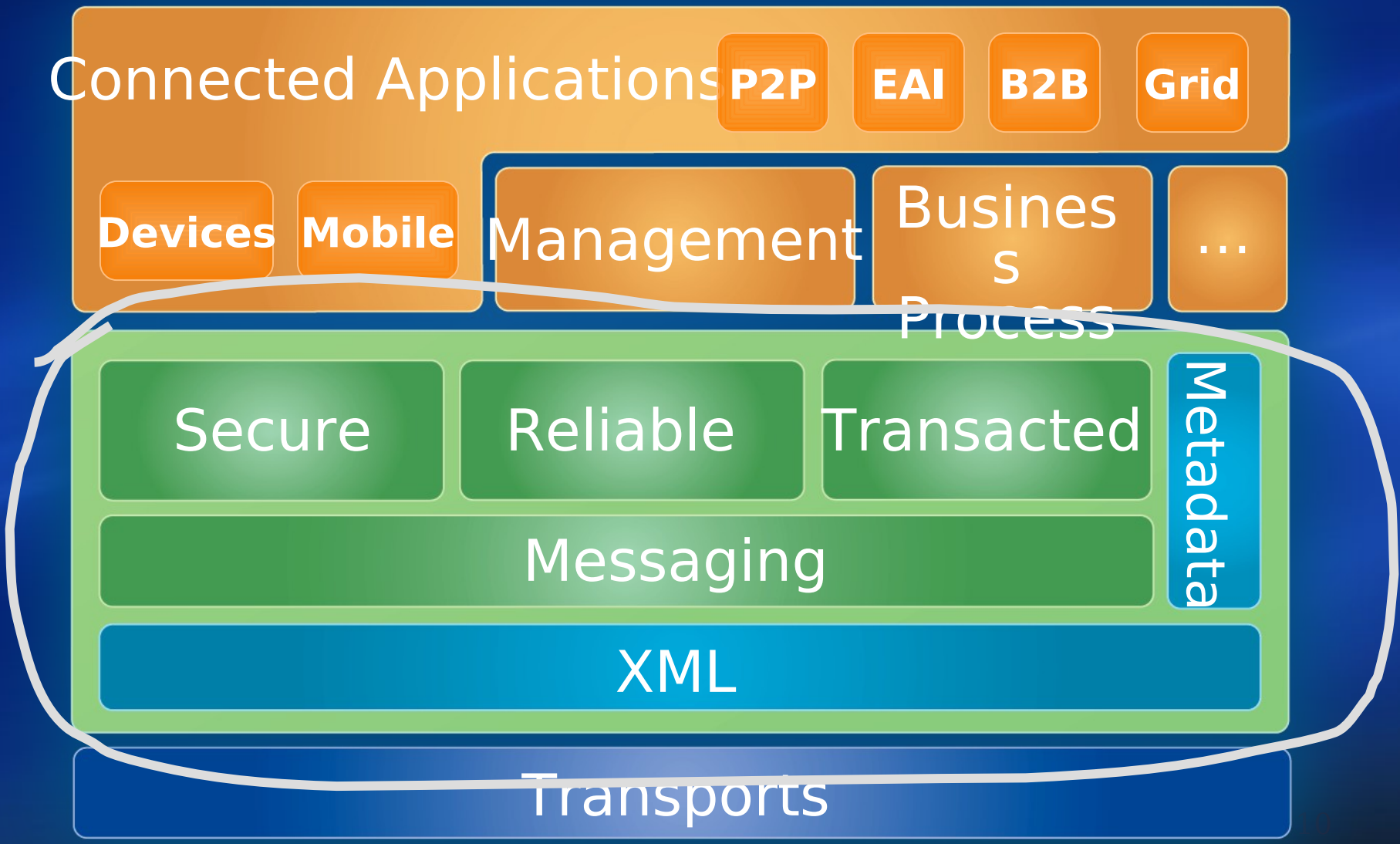


Assumes policies are known/cached

Backend Processing



Spec Drilldown



Web Services Design Principles

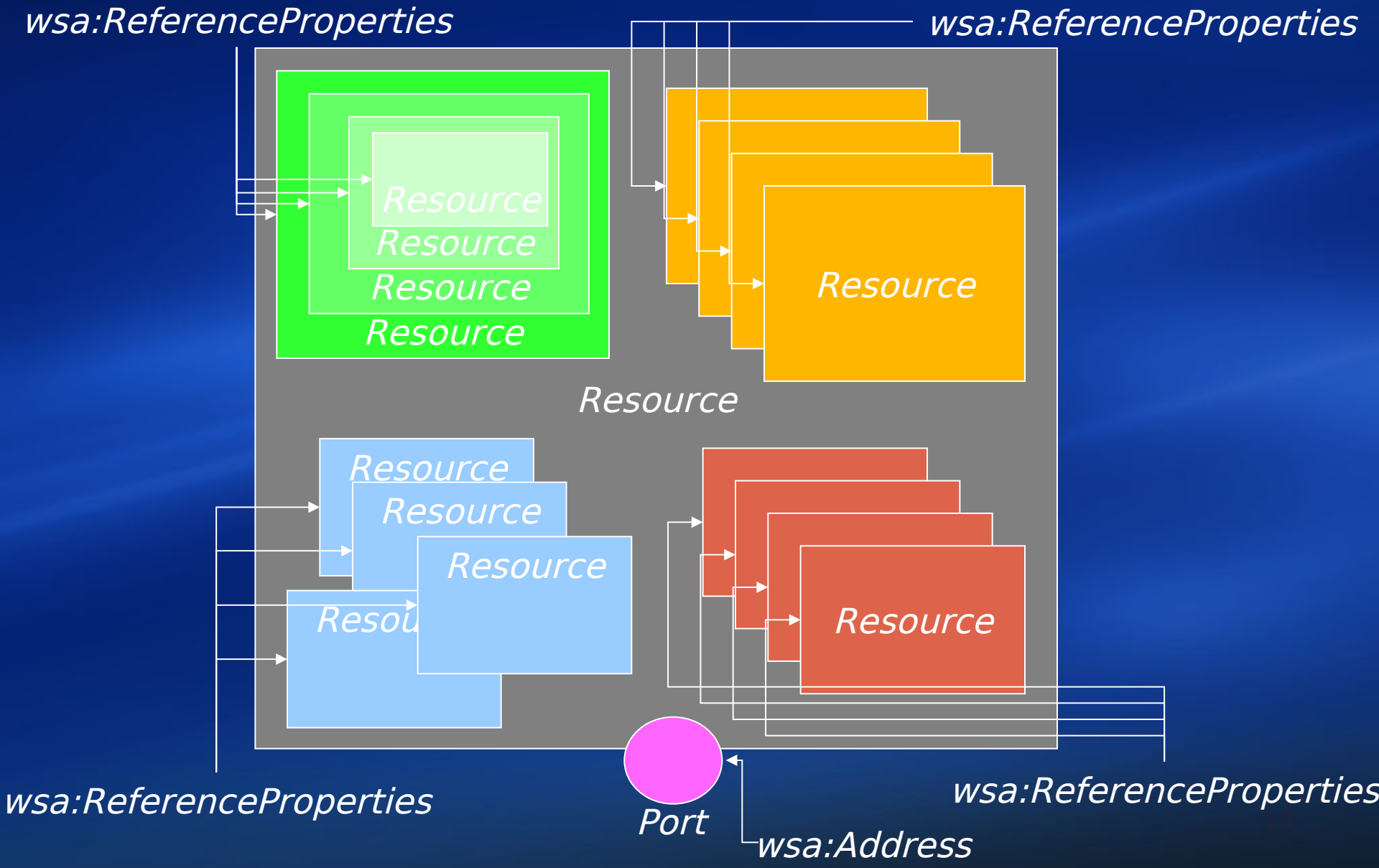
- A **general-purpose, composable** protocol framework
- Architecture is **metadata-driven, policy-based**
- Protocols are **factored** from both transports and application semantics
- Architecture allows **intermediaries**
- Uniform **data model** for protocols/content
- Broad industry partnership

Messaging

WS-Addressing

- WS-Addressing provides mechanisms for addressing resources
  - Shipping those addresses in messages
  - Addressing messages to those resources
- Addressing mechanisms are transport-neutral
- Resources are not constrained
  - Can be constructed, partitioned, named, addressed in arbitrary fashion
- Internal resource organization is opaque

Resource Model



Endpoint Reference

- "Generalized URI"
  - EPR = Address + Reference Properties

```
<a:EndpointReference
  xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'
  xmlns:m='http://example.net/ws/weather/forecasts'
  xmlns:p='http://schemas.xmlsoap.org/ws/2002/12/policy' >
  <a:Address>http://example.org/weather/us</a:Address>
  <a:ReferenceProperties>
   <m:Info>Services.Weather.Wind</m:Info>
  </a:ReferenceProperties>
  <p:Policy>
  . . .
  </p:Policy>
<a:/EndpointReference>
```

[Address] property -
Network address of resource

[Reference Properties] -
Other addressing information

Policy for this resource

Message Information Headers

- WS-Addressing also defines header elements

Property	Header	Notes
destination	***wsa:To***	From wsa:Address
recipient	wsa:Recipient	An endpoint reference
source endpoint	wsa:From	An endpoint reference
reply endpoint	wsa:ReplyTo	An endpoint reference
fault endpoint	wsa:FaultTo	An endpoint reference
action	***wsa:Action***	Name of logical action
message id	wsa:MessageID	Unique ID
relationship	wsa:RelatesTo	Provides for correlation
*	*	From wsa:ReferenceProperties

Request/Response With WSA

```
<s:Envelope xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'>
  <s:Header>
   <a:To>http://example.org/weather/us</a:To>
   <a:Action>http://example.org/weather/forecast</wsa:Action>
   <a:ReplyTo>. . .</a:ReplyTo>
  </s:Header>
  <s:Body> . . . </s:Body>
<s:/Envelope>
```

Request

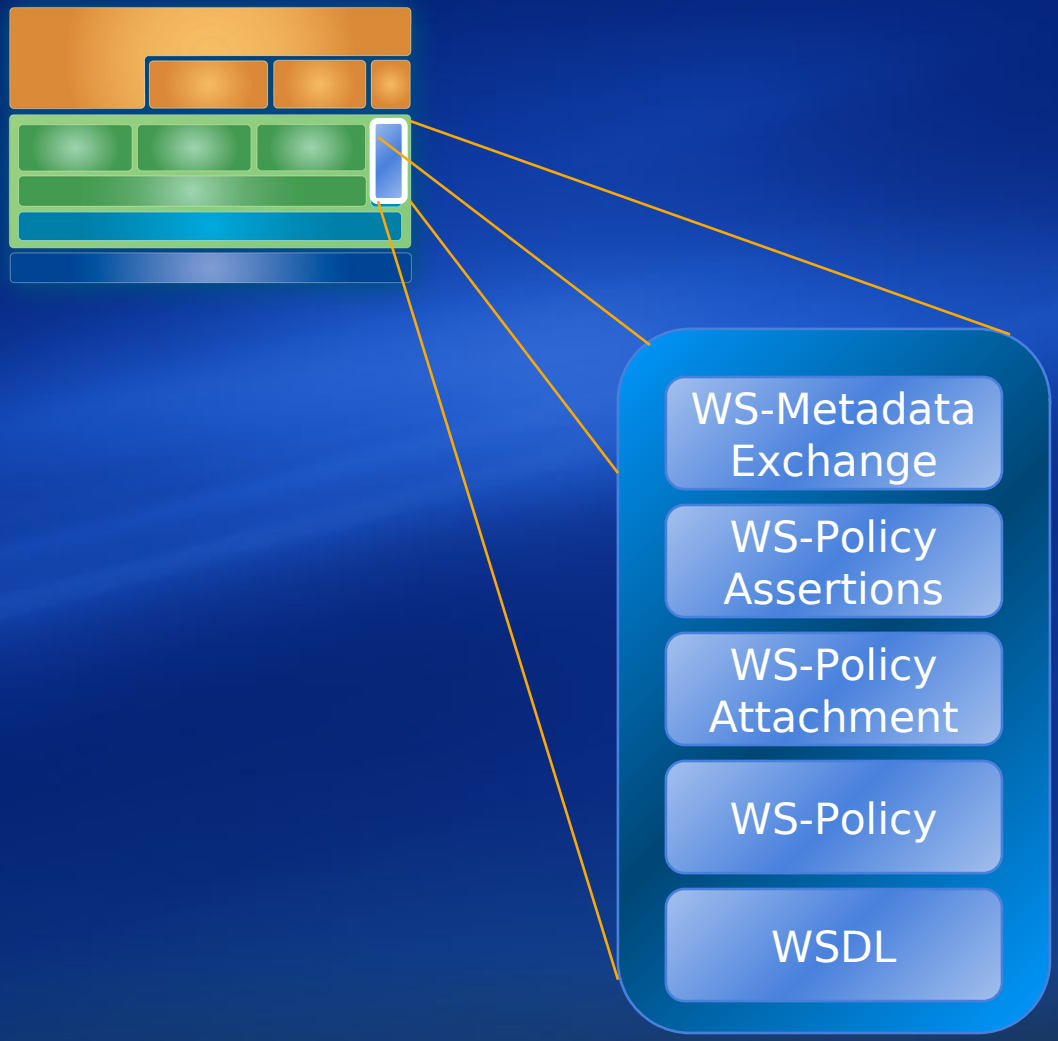
SOAP 1.2

```
<s:Envelope xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'>
  <s:Header>
   <a:To>http://example.org/weatherrequestor</a:To>
   <a:Action>http://example.org/weather/forecastR</wsa:Action>
   <a:RelatesTo RelationshipType='wsa:Response' >
   . . .
   </a:RelatesTo>
  </s:Header>
  <s:Body> . . . </s:Body>
<s:/Envelope>
```

Response

SOAP 1.2

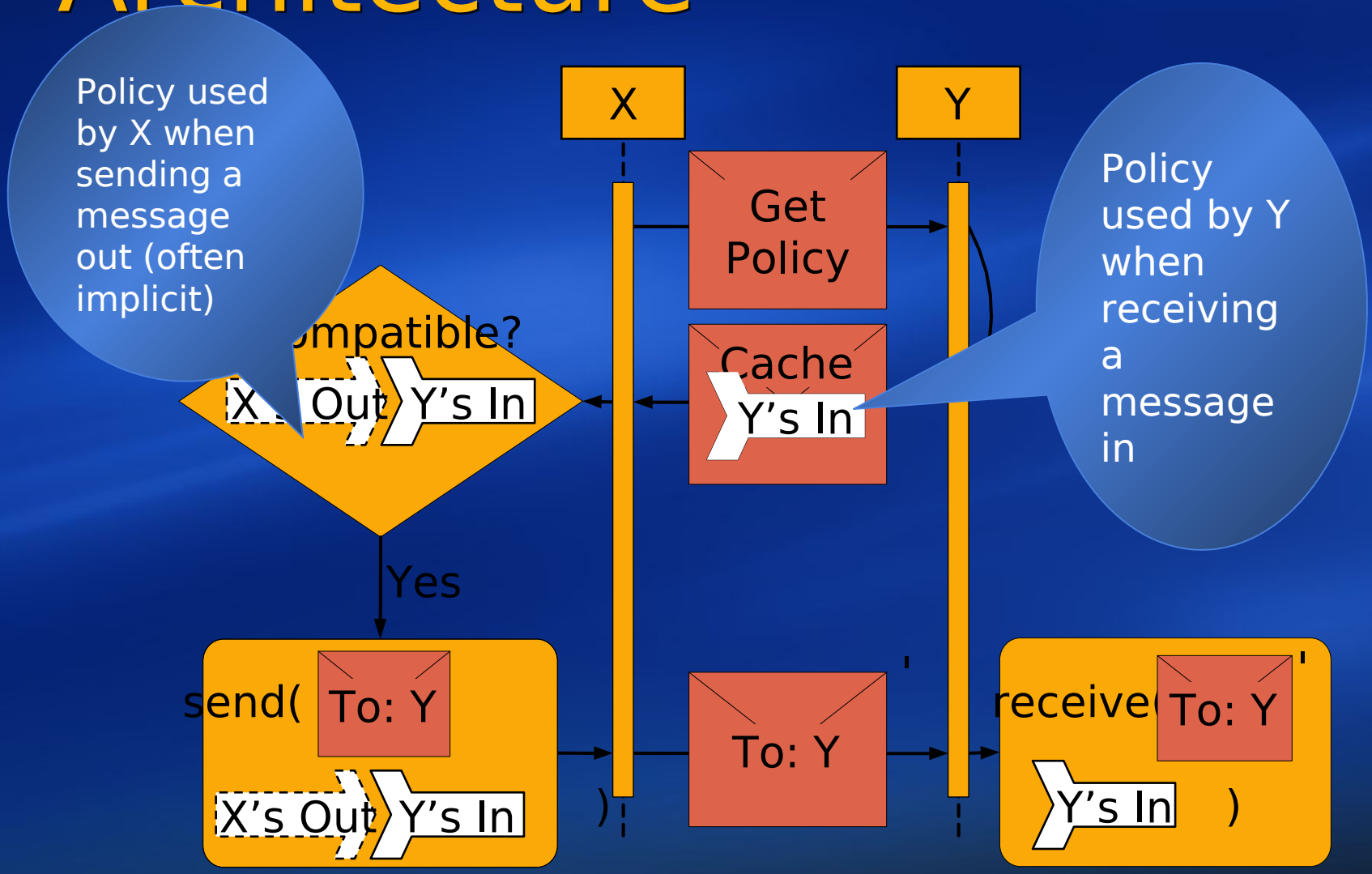
Metadata


WS-Metadata Exchange
WS-Policy Assertions
WS-Policy Attachment
WS-Policy
WSDL

Policy

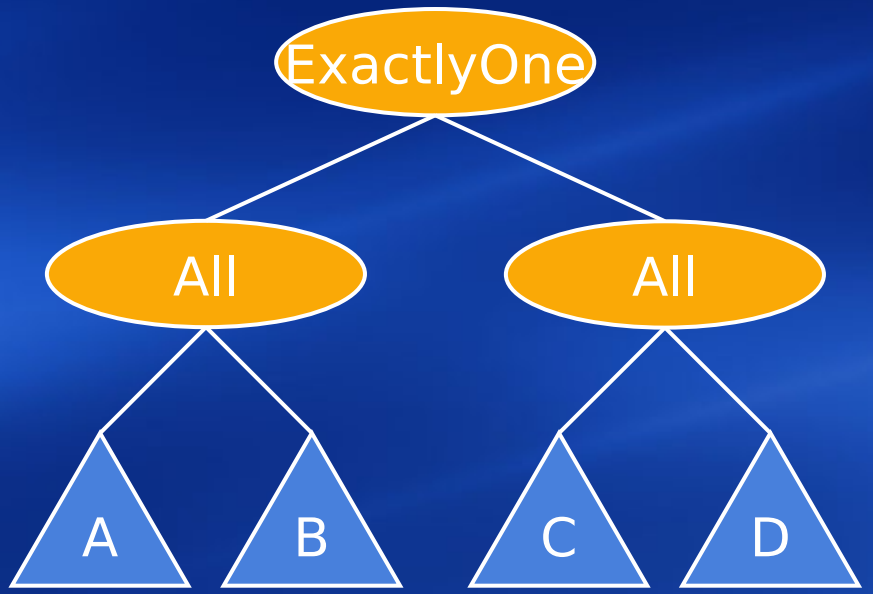
- WS-Policy: A framework for making statements about resources
  - Used to express receiver requirements for incoming messages (e.g., transports, security)
  - At runtime, can be used to match requirements to capabilities
- WS-PolicyAssertions: Predefined basics
- WS-PolicyAttachment: Attaching policy expression to a subject (e.g., EPR)

Metadata Driven Architecture



WS-Policy

- Policy expressions express choices over domain-specific assertions
- Operators and usages allow writing policies in a compact form
- Preferences are hints in choosing between alternatives



WS-Policy Example

```
<wsp:ExactlyOne wsp:Usage='wsp:Required' >
 <wsp:All wsp:Preference='1'>
  <jeff:MainCourse>Burger</jeff:MainCourse>
  <jeff:Side>Fries</jeff:Side>
 </wsp:All>
 <wsp:All wsp:Preference='2'>
  <jeff:MainCourse>Pizza</jeff:MainCourse>
  <jeff:Side>Yogurt</jeff:Side>
 </wsp:All>
</wsp:ExactlyOne>
```

Preference

Domain-specific assertions

Usage

Operators

WS-PolicyAssertions

- TextEncoding: Character encoding

```
<wsp:TextEncoding Encoding='utf-8' />
```

- Language: Natural (human) language

```
<wsp:Language Language='en-US' />
```

- SpecVersion: URI asserting compliance with spec

```
<wsp:SpecVersion wsp:Usage='wsp:Required'
  URI='http://schemas.xmlsoap.org/ws/2002/04/secext' />
```

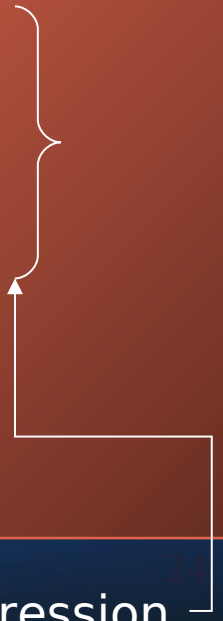
- MessagePredicate: XPath (or other) statement about message

```
<wsp:MessagePredicate wsp:Usage='wsp:Required' >
 count(wsp:GetHeader(.)/wsse:Security) = 1
</wsp:MessagePredicate>
```

WS-PolicyAttachment

- Defines what a policy applies to
- Associates domain expression with policy
  - Domain expression indicates scope
  - Policy makes statements about that scope

```
<wsp:PolicyAttachment>
 <wsp:AppliesTo>
  <wsa:EndpointReference>
   <wsa:Address>http://jcs.org/lunch</wsa:Address>
  </wsa:EndpointReference>
 </wsp:AppliesTo>
 <wsp:Policy> . . . </wsp:Policy>
</wsp:PolicyAttachment>
```



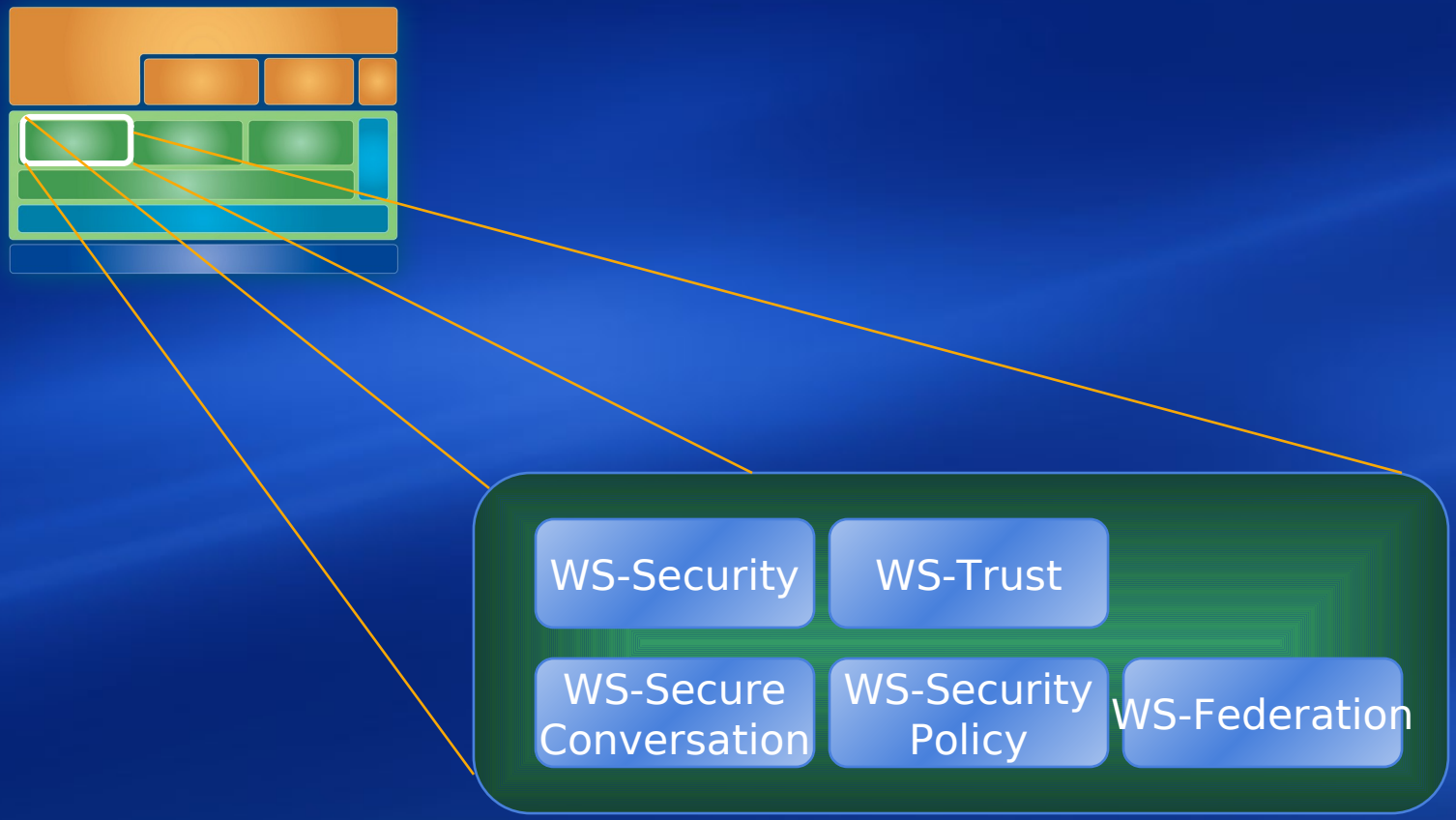
WS-MetadataExchange

- How do I get metadata for another endpoint anyway?
  - Extracted from WSDL
  - Returned from discovery FIND (e.g., UDDI)
  - Out of band
  - MetadataExchange: Retrieved by a simple GET
    - Applies to Policy, WSDL, Schema

```
<s:Envelope>
 <s:Header>
  <wsa:Action>
  http://schemas.xmlsoap.org/ws/2003/10/md-exchange/GetPolicy
  </wsa:Action>
 </s:Header>
 <s:Body />
</s:Envelope>
```

[Conceptual]

Security


WS-Security
WS-Trust
WS-Secure Conversation
WS-Security Policy
WS-Federation

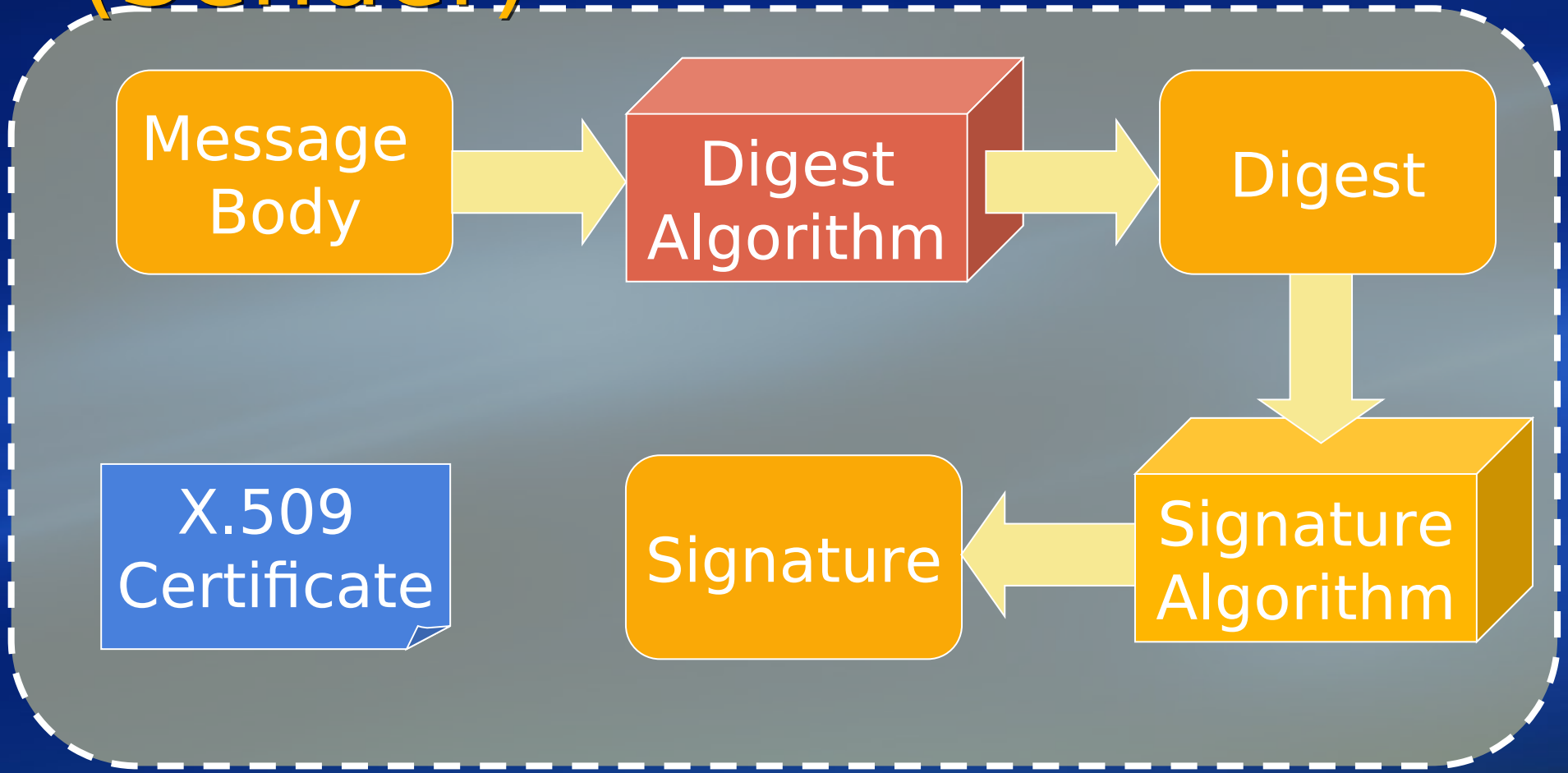
WS-Security

- Defines a framework for building security protocols
  - Integrity
  - Confidentiality
  - Propagation of security tokens
- Framework designed for end-to-end security of SOAP messages
  - From initial sender, through 0-n intermediaries to ultimate receiver
- Support for pluggable algorithms
  - Encryption, Digest, Signature, Canonicalization, Transforms

Integrity Example (Sender)

- I want to send a SOAP message and ensure that the body is not modified
  - I generate a digest of the SOAP body
  - I generate a signature over the digest (and some other info) using my private key
  - I include my certificate (which includes my public key) in a security token

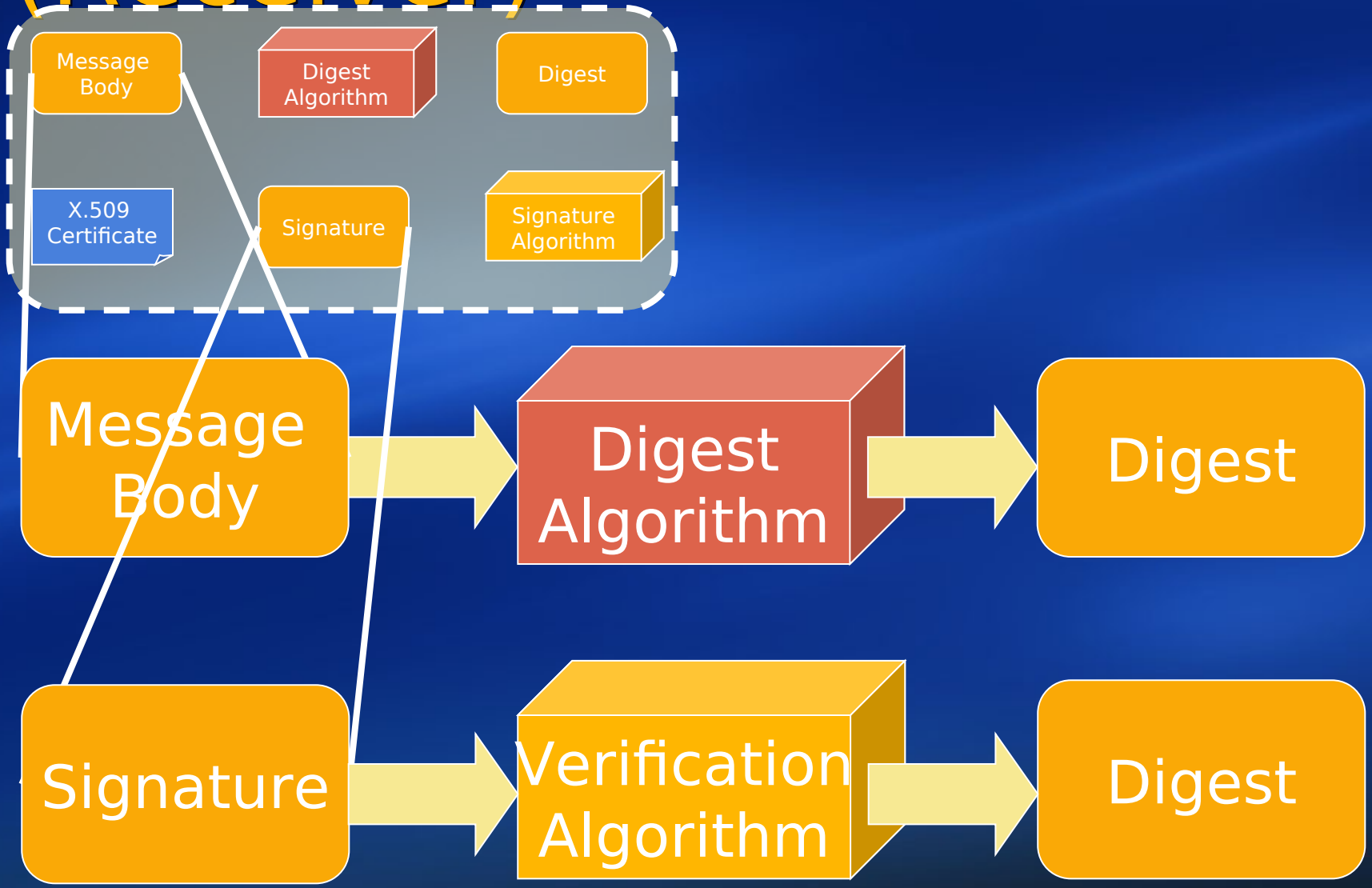
Integrity Example (Sender)



Integrity Example (Receiver)

- You want to check that the body of the message was not modified
  - You generate a digest for the SOAP body
  - You verify the signature using my public key (which you got from my certificate)
  - You compare the two values and ensure they match
- As a side-effect, you also know the message was from me

Integrity Example (Receiver)



```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext'
  xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
 <s:Header>
  <ws:Security s:mustUnderstand='true' >
   <ws:BinarySecurityToken wsu:Id='Me' ValueType='ws:X509v3'
                           EncodingType='ws:Base64Binary' >
   MeIIZFgea4FGiu5cvWEklO8pl...
   </ws:BinarySecurityToken>
   . . .
```

My security token

```
  . . .
  <ds:Signature>
   <ds:SignedInfo>
    <ds:CanonicalizationMethod
Algorithm='http://www.w3.org/2001/10/xml-exc-c14n#' />
    <ds:SignatureMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
    <ds:Reference URI='#Body' >
      <ds:DigestMethod
Algorithm='http://www.w3.org/2000/09/xmldsig#sha1' />
     <ds:DigestValue>uJhGtef54ed91iKLoA...</ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>FR8yaKmNDePQ7E3Hj...</ds:SignatureValue>
   . . .
```

Reference to data I want to protect

Digest of data I want to protect

Signature over ds:SignedInfo element

```
    . . .
    <ds:KeyInfo>
     <ws:SecurityTokenReference>
      <ws:Reference URI='#Me' />
     </ws:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
  </ws:Security>
  . . .
 </s:Header>
 <s:Body wsu:Id='Body' >
  . . .
 </s:Body>
</s:Envelope>
```

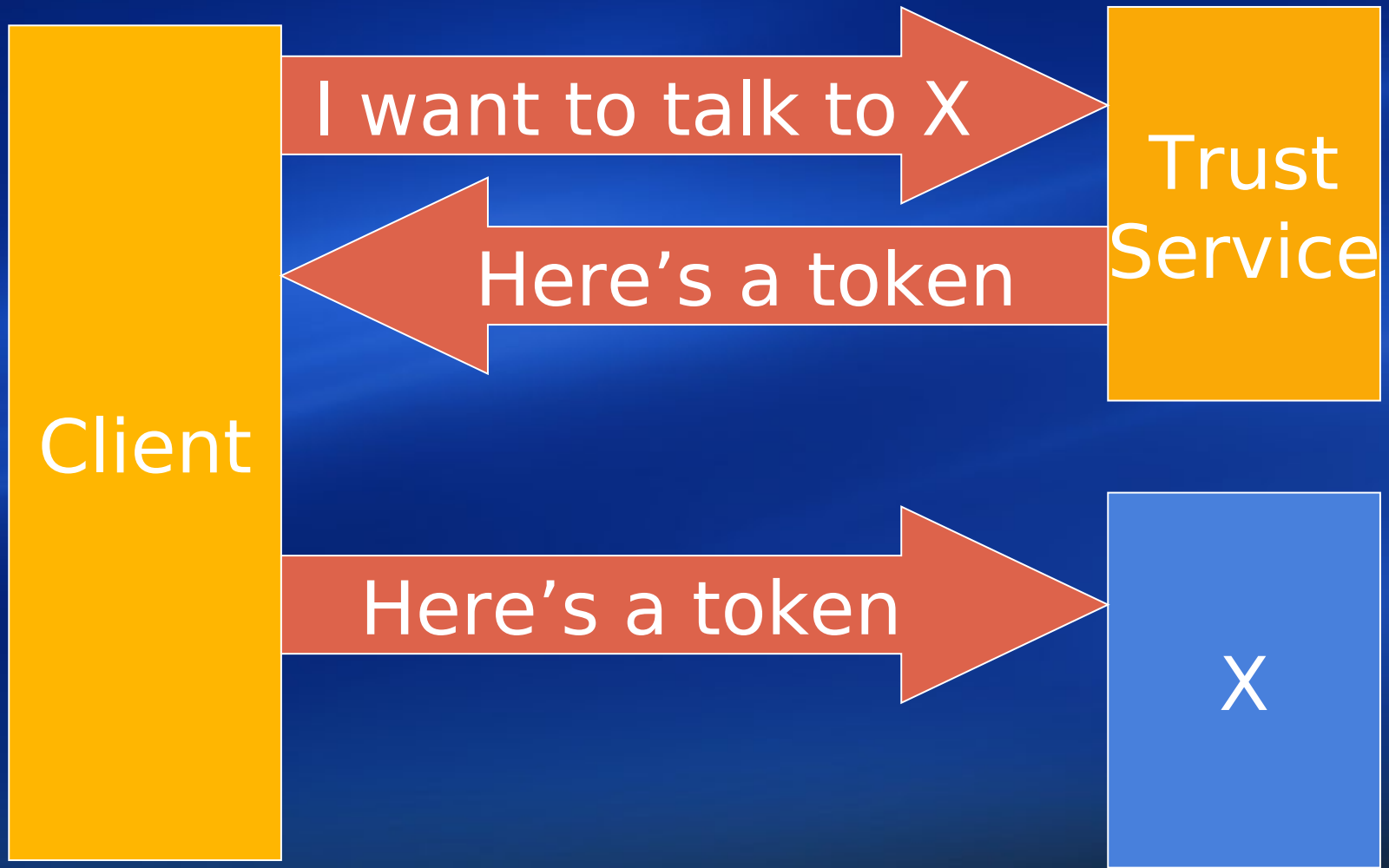
Reference to certificate that can be used to verify signature

Signed data

WS-Trust

- Defines how to broker trust relationships
  - Some trust relationship has to exist a priori
- Defines how to exchange security tokens
- Defined as an interface specification for a Security Token Service
  - A RequestSecurityToken message is sent to the trust service
  - It responds with a RequestSecurityTokenResponse

Example


Client
I want to talk to X
Here's a token
Trust Service
Here's a token
X

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext'
  xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
 <s:Header>
  <ws:Security s:mustUnderstand='true' >
    <ws:BinarySecurityToken wsu:Id='Me'
                            ValueType='ws:Kerberosv5TGT'
                            EncodingType='ws:Base64Binary' >
   MehgVCD3Efg7hJlaFvCdEFujH...
   </ws:BinarySecurityToken>
   <ds:Signature>
    <!-- Signature over request -->
   </ds:Signature>
  </ws:Security>
 </s:Header>
 . . .
```

Security token that I share with the trust service

```
 . . .
 <s:Body wsu:Id='request' >
  <ws:RequestSecurityToken>
   <ws:TokenType>ws:Kerberosv5ST</ws:TokenType>
   <ws:RequestType>ws:ReqIssue</ws:RequestType>
   <ws:Base><ws:Reference URI='#Me' /></ws:Base>
  </ws:RequestSecurityToken>
 </s:Body>
</s:Envelope>
```

Type of token I want

I want a new token

Reference to security token I share with the STS

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext' >
 <s:Header>
  <ws:Security s:mustUnderstand='true' >
   <ws:BinarySecurityToken wsu:Id='Me'
                           ValueType='ws:Kerberosv5TGT'
                           EncodingType='ws:Base64Binary' >
   MehgVCD3Efg7hJlaFvCdEFujH...
   </ws:BinarySecurityToken>
  </ws:Security>
 </s:Header>
 <s:Body wsu:Id='response' >
  <ws:RequestSecurityTokenResponse>
   <ws:RequestedSecurityToken>
    <ws:BinarySecurityToken ValueType='ws:Kerberosv5ST'
                            EncodingType='ws:Base64Binary' >
    AlhgVCD3Efg7hJlaFvCdEFujH...
    </ws:BinarySecurityToken>
   </ws:RequestedSecurityToken>
   . . .
```

Requested security token

Security token that I share with the trust service

```
  . . .
  <ws:RequestedProofToken>
   <xe:EncryptedKey Id='Sym' >
    <xe:EncryptionMethod
Algorithm='http://www.w3.org/2001/04/xmlenc#des' />
    <ds:KeyInfo>
     <ws:SecurityTokenReference>
      <ws:Reference URI='#Me' />
     </ws:SecurityTokenReference>
   </ds:KeyInfo>
   <xe:CipherData>
     <xe:CipherValue>bvDfEg6Sh7GbCvDiAl</xe:CipherValue>
   </xe:CipherData>
   <xe:EncryptedKey>
  </ws:RequestedProofToken>
 </ws:RequestSecurityTokenResponse>
</s:Body>
</s:Envelope>
```

Requested proof token

Reference to security token that I share with the trust service

WS-SecureConversation

- WS-Security provides for single message security
- Nodes will often want to exchange more than one message
  - Specifying new symmetric keys for each message is tedious and verbose
- WS-SecureConversation defines mechanisms to address this

WS-SecureConversation

- Participants establish a shared context
  - Context contains keys and other information
  - Has an identifier – used in subsequent messages
  - Optionally has creation/expiry timestamps
- Context can be established in a variety of ways
  - Using WS-Trust

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext'
  xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
 <s:Header>
 <ws:Security s:mustUnderstand='true' >
  <ws:SecurityContextToken>
   <wsu:Identifier>
   uuid:652d2aaa-4857-4d8c-865c-f9549e5806f0</wsu:Identifier>
   <wsu:Created>2003-07-13T18:17:44Z</wsu:Created>
   <wsu:Expires>2003-07-13T19:17:44Z</wsu:Expires>
   <ws:Keys>
    <xenc:EncryptedKey>
    . . .
    </xenc:EncryptedKey>
   </ws:Keys>
  </ws:SecurityContextToken>
 </ws:Security>
 </s:Header>
 <s:Body wsu:Id='request'>
 . . .
 </s:Body>
</s:Envelope>
```

Subsequent Exchanges

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext'
  xmlns:ds='http://www.w3.org/2000/09/xmldsig#' >
 <s:Header>
 <ws:Security s:mustUnderstand='true' >
  <ws:SecurityContextToken>
   <wsu:Identifier>
   uuid:652d2aaa-4857-4d8c-865c-f9549e5806f0</wsu:Identifier>
  </ws:SecurityContextToken>
 </ws:Security>
 </s:Header>
 <s:Body wsu:Id='response'>
 . . .
 </s:Body>
</s:Envelope>
```

WS-SecurityPolicy

- A set of policy assertions related to concepts defined by other WS-Sec* specs
- Allows participants to specify
  - Token types
  - Whether integrity and/or confidentiality are required
  - Algorithms for the above
  - Which message parts need signing/encrypting

Security Assertion Examples

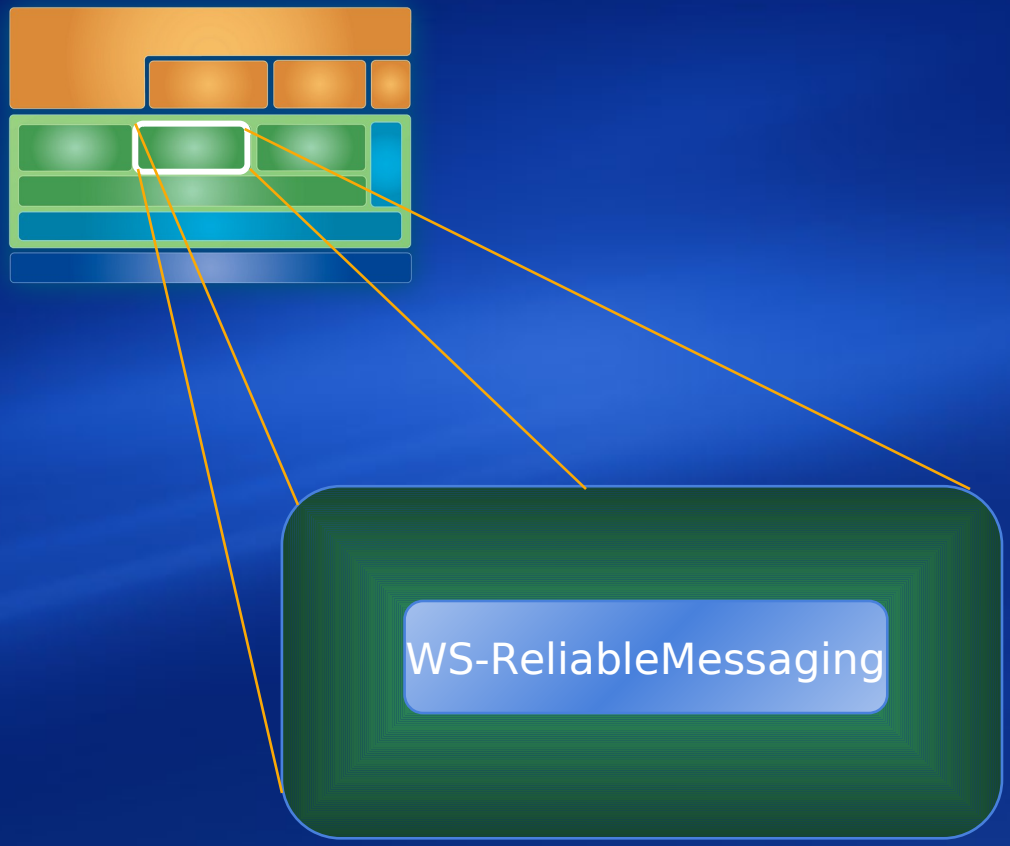
```
<wsp:ExactlyOne wsp:Usage='Required'
    xmlns:wsp='http://schemas.xmlsoap.org/ws/2002/12/policy' >
 <ws:SecurityToken wsp:Preference='100'
    xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext' >
  <ws:TokenType>ws:x509v3</ws:TokenType>
 </ws:SecurityToken>
 <ws:SecurityToken wsp:Preference='1'
    xmlns:ws='http://schemas.xmlsoap.org/ws/2003/07/secext' >
  <ws:TokenType>ws:Kerberosv5TGT</ws:TokenType>
 </ws:SecurityToken>
</wsp:ExactlyOne>
```

Security Token

```
<ws:Integrity wsp:Usage='Required'
 <ws:Algorithm Type='ws:AlgCanonicalization'
          URI='http://www.w3.org/.../xml-exc-c14n' />
 <ws:Algorithm Type='ws:AlgSignature'
          URI='http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
 <ws:Algorithm Type='ws:AlgDigest'
          URI='http://www.w3.org/2000/09/xmldsig#hmac-sha1' />
</ws:Integrity>
```

Integrity

Reliable Messaging



Distributed State Coordination

- Reliable Messaging
  - Two-party (source, destination), asynchronous
  - Exactly once, in-order
- Atomic Transaction
  - Multi-party, all or nothing state change, synchronous
  - Two Phase Commit
    - Phase 1: Prepare to Commit
    - Phase 2: Commit or Abort
- Business Activity
  - Multi-party, final consistent state, asynchronous
  - Two Phases (almost)
    - Phase 1: Cancel/Complete
    - Phase 2: Close/Compensate
    - Anytime: Exit/Fault

WS-ReliableMessaging

- RM defines QoS over unidirectional message sequences
  - Exactly once, in-order
- Simplifies application programming – no need to defend against
  - Lost, duplicated or delayed messages
  - Endpoint failure
- Acknowledgements sent upon receipt
  - As opposed to after application processing
- Durability is orthogonal to protocol definition

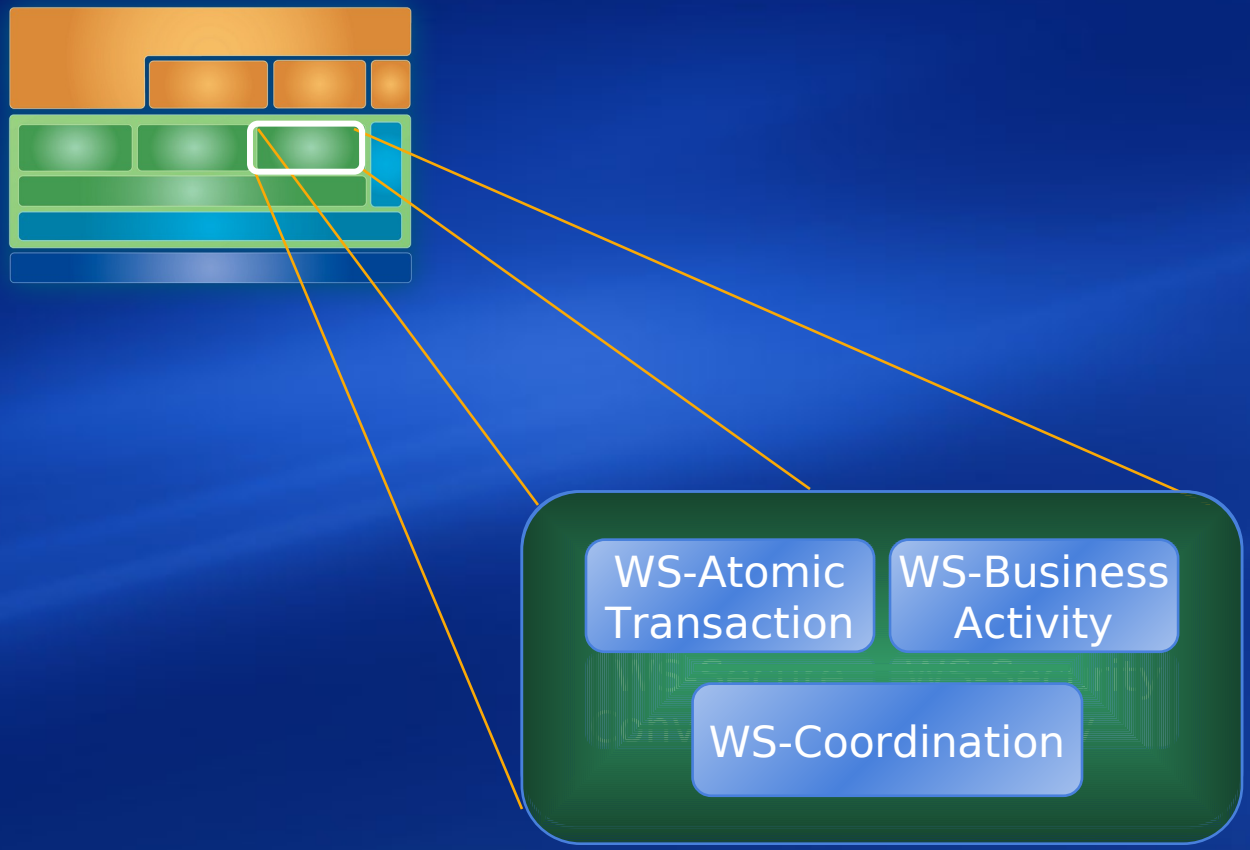
Sequence Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wsrm='http://schemas.xmlsoap.org/ws/2003/03/wsrm‘ >
 <s:Header>
  <wsa:Action>CreateCoordinationContext</wsa:Action>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
  <wsrm:Sequence>
    <wsu:Identifier>
      http://fabrikam123.com/abc
    </wsu:Identifier>
    <wsrm:MessageNumber>10</wsrm:MessageNumber>
    <wsrm:LastMessage/>
  </wsrm:Sequence>
 </s:Header>
 <s:Body> . . . </s:Body>
</s:Envelope>
```

Acknowledgement Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wsrm='http://schemas.xmlsoap.org/ws/2003/03/wsrm‘ >
 <s:Header>
  <wsa:Action>CreateCoordinationContext</wsa:Action>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
  <wsrm:SequenceAcknowledgement>
    <wsu:Identifier>
      http://fabrikam123.com/abc
    </wsu:Identifier>
    <wsrm:AcknowledgementRange Upper=’10’ Lower=‘1’ />
  </wsrm:SequenceAcknowledgement>
 </s:Header>
 <s:Body> . . . </s:Body>
</s:Envelope>
```

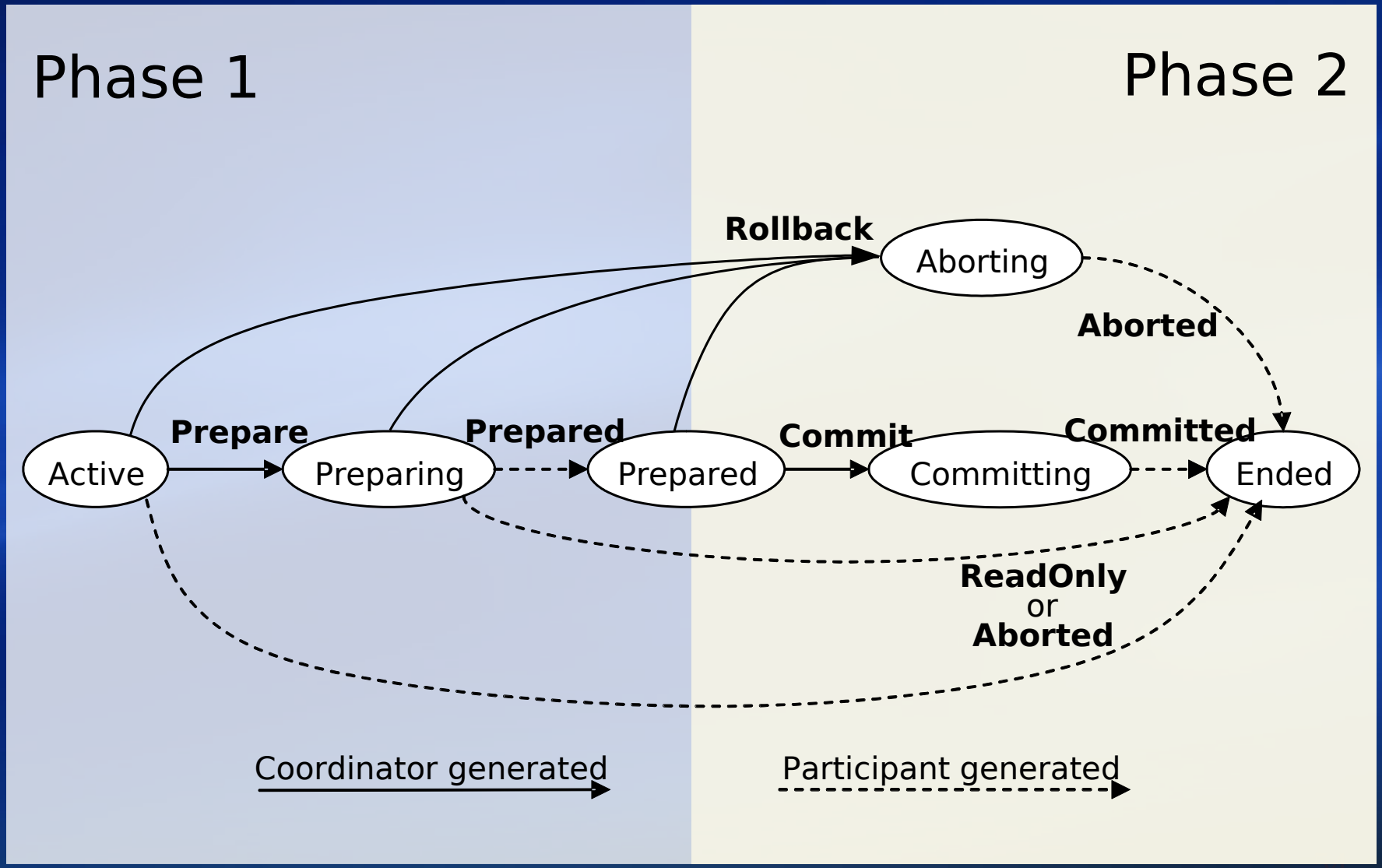
Transactions



WS-AtomicTransaction

- Classic 2 Phase Commit ACID protocol
  - Prepare
  - Commit/Rollback
- Durable or volatile
- All resources must be up (synchronous)
- All-or-nothing (complete agreement)
- Uses asynchronous messages
- Resources are locked – easy programming model
- Appropriate for scenarios with shared assumptions about latency/trust

AT Abstract State Diagram



WS-BusinessActivity

- Looser isolation model
  - Intermediate states visible
  - Operations cannot be undone
  - Shared state “settles out” at end of interaction
- Tries to move forward to a known good state
  - May try “Plan B”
  - May use compensation
- Relaxes “complete agreement” requirement
- No requirement to lock resources
- Appropriate for scenarios crossing trust domains

WS-Coordination

- Base Protocol for AT and BA
- Creates and Registers for Transactions
- Generates Coordination Context
  - Passes Data to Register for Transactions
- Based on WS-Addressing
- Extensible Coordination Types
  - AT and BA are the two predefined ones

WS-AT Register Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wscoor='http://schemas.xmlsoap.org/ws/2003/09/wscoor‘ >
 <s:Header>
  <wsa:Action>http://…/2003/09/wscoor#Register
  </wsa:Action>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
 </s:Header>
 <s:Body>
  <wscoor:Register>
   <wscoor:ProtocolIdentifier>
     http://schemas.xmlsoap.org/ws/2003/09/wsat#Volatile2PC
      </wscoor:ProtocolIdentifier>
   <wscoor:ParticipantProtocolService>
    <wsa:Address>http://localhost/Participant1</wsa:Address>
   </wscoor:ParticipantProtocolService>
  </wscoor:Register>
 </s:Body>
</s:Envelope>
```

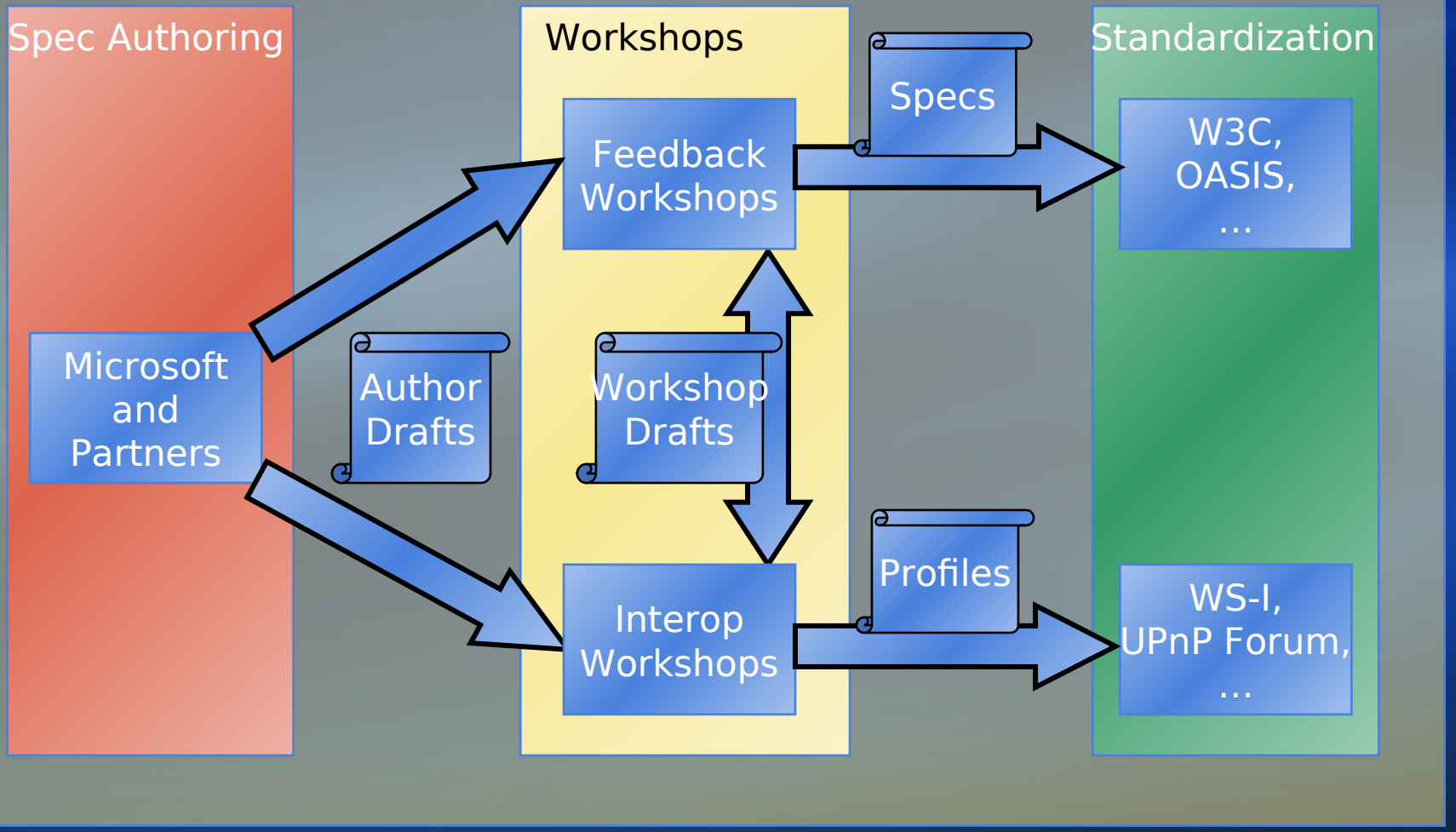
WS-AT Register Response

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wscoor='http://schemas.xmlsoap.org/ws/2003/09/wscoor'
  xmlns:mstx='http://schemas.microsoft.com/wsat/extensibility'>
 <s:Header>
  <wsa:To>http://localhost/Participant1</wsa:To>
  <wsa:Action>http://…/wscoor#RegisterResponse</wsa:Action>
 </s:Header>
 <s:Body>
  <wscoor:RegisterResponse>
   <wscoor:CoordinatorProtocolService>
    <wsa:Address>http://localhost/TestCoordinator</wsa:Address>
    <wsa:ReferenceProperties>
     <mstx:transactionId>urn:3fd333f8-…</mstx:transactionId>
     <mstx:enlistmentId>urn:d79c54be-…</mstx:enlistmentId>
    </wsa:ReferenceProperties>
   </wscoor:CoordinatorProtocolService>
  </wscoor:RegisterResponse>
 </s:Body>
</s:Envelope>
```

WS-AT Prepare Example

```
<s:Envelope
  xmlns:s='http://www.w3.org/2003/05/soap-envelope'
  xmlns:wsu='http://schemas.xmlsoap.org/ws/2002/07/utility'
  xmlns:wscoor='http://schemas.xmlsoap.org/ws/2003/09/wscoor'
  xmlns:wsat='http://schemas.xmlsoap.org/ws/2003/09/wsat'
  xmlns:mstx='http://schemas.microsoft.com/wsat/extensibility'>
 <s:Header>
  <wsa:To>http://localhost/TestCoordinator</wsa:To>
  <wsa:Action>http://…/2003/09/wsat#Prepare</wsa:Action>
  <mstx:transactionId>urn:3fd333f8-…</mstx:transactionId>
  <mstx:enlistmentId>urn:d1de41df-…</mstx:enlistmentId>
 </s:Header>
 <s:Body>
  <wsat:Prepare />
 </s:Body>
</s:Envelope>
```

Spec Development Process



Call To Action

- Further Resources
  - PDC CD has spec pointers and narrated powerpoint tutorials by spec authors
  - http://msdn.microsoft.com/webservices
  - omrig@microsoft.com
- Early adopter enterprises: Use WSE
  - Yearly RTM of latest advanced WS specs
  - 2+1 support policy
  - No backward/forward compatibility guarantees
- Attend Workshops
  - http://msdn.microsoft.com/webservices/community/workshops
  - Give feedback on specs
  - Interoperate with other implementations

For More Information

- Come see me
  - Immediately after this session
  - Web/Services Lounge: 309 Foyer
- MSDN “Longhorn” DevCenter http://msdn.microsoft.com/longhorn
- Newsgroup
  - microsoft.public.windows.developer.winfx.indigo
- At PDC
  - Hands on labs: On-site or download from CommNet
  - Ask The Experts: Tuesday 7 P.M. – 9 P.M. Hall G, H
  - PDC Weblogs: http://pdcbloggers.net

Microsoft®

PDC03

Make the connection